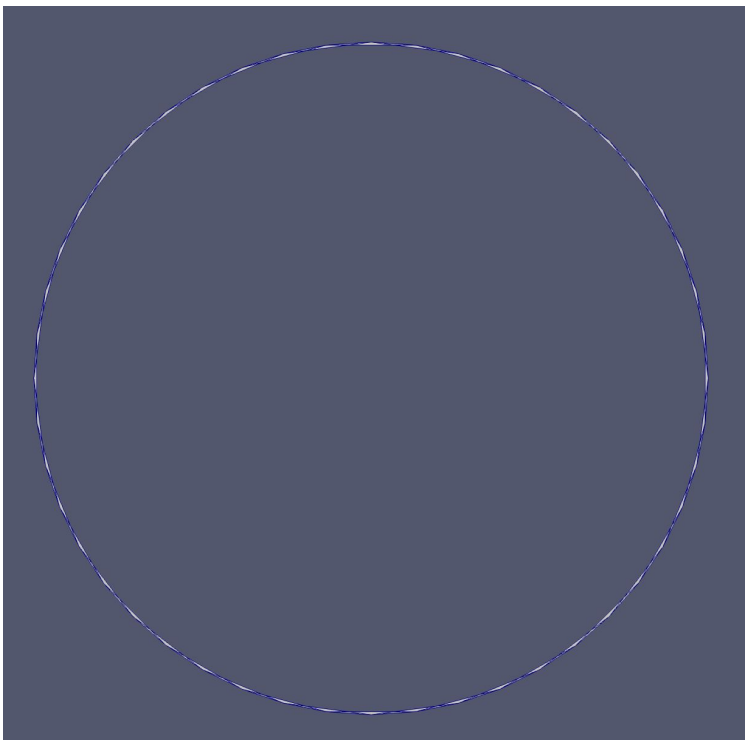
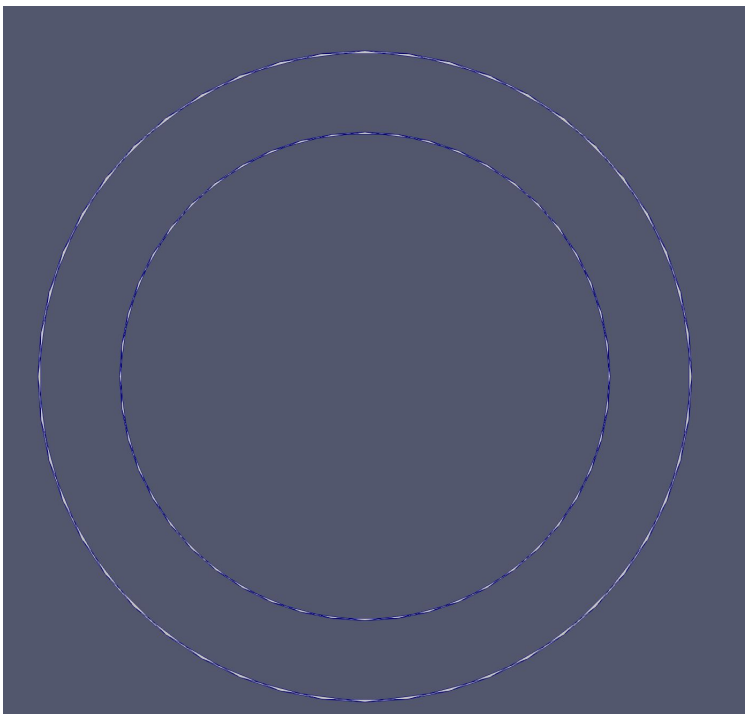
Andrea Padula

REPORT

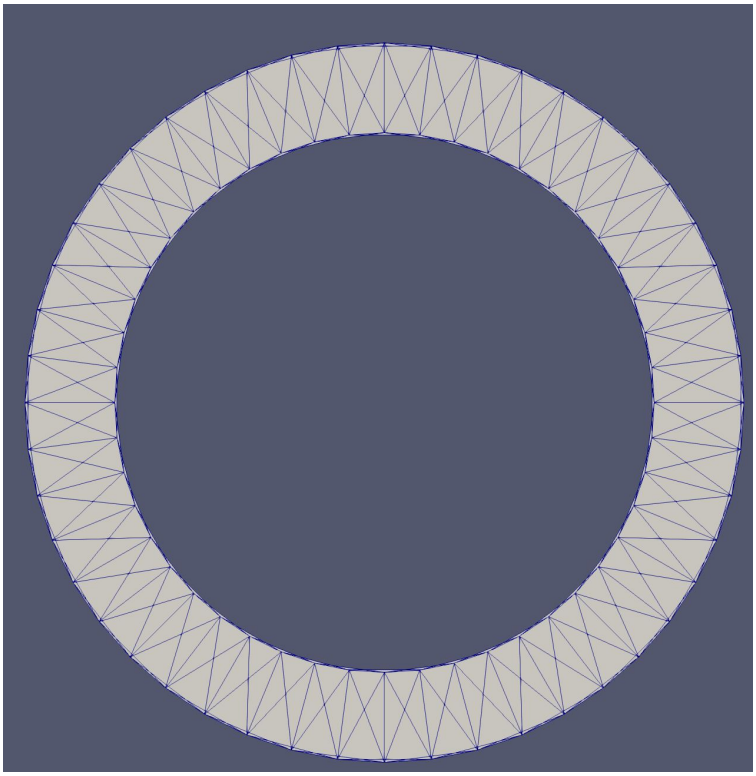
The first experiment that I did was the one using the annulus_small.vtp. To reconstruct the annulus I set the radius very small like 0.01 and kept increasing until I saw something showing up in my screen. At 0.08 a circle appeared



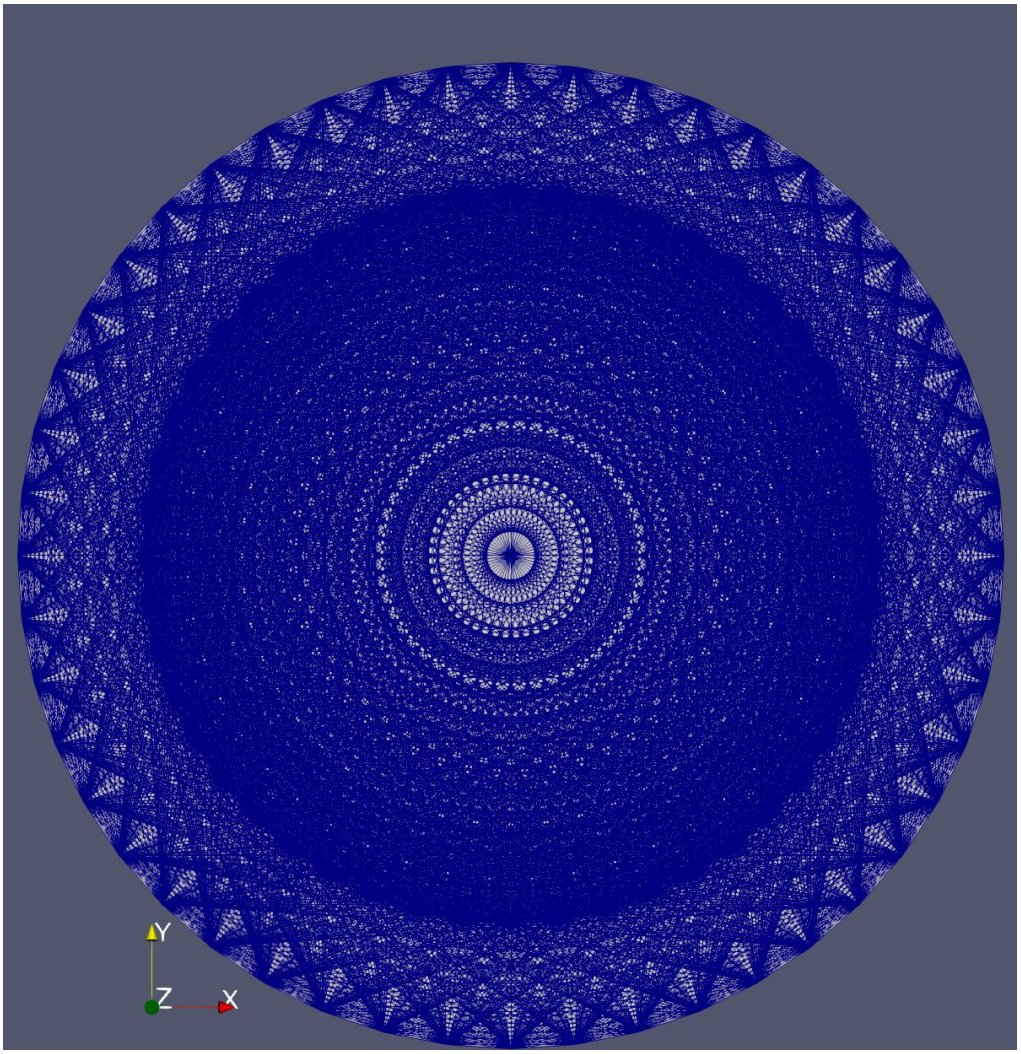
After by increasing the size of my radius to 0.12



After I used 0.14 and finally the shape of annulus showed up.

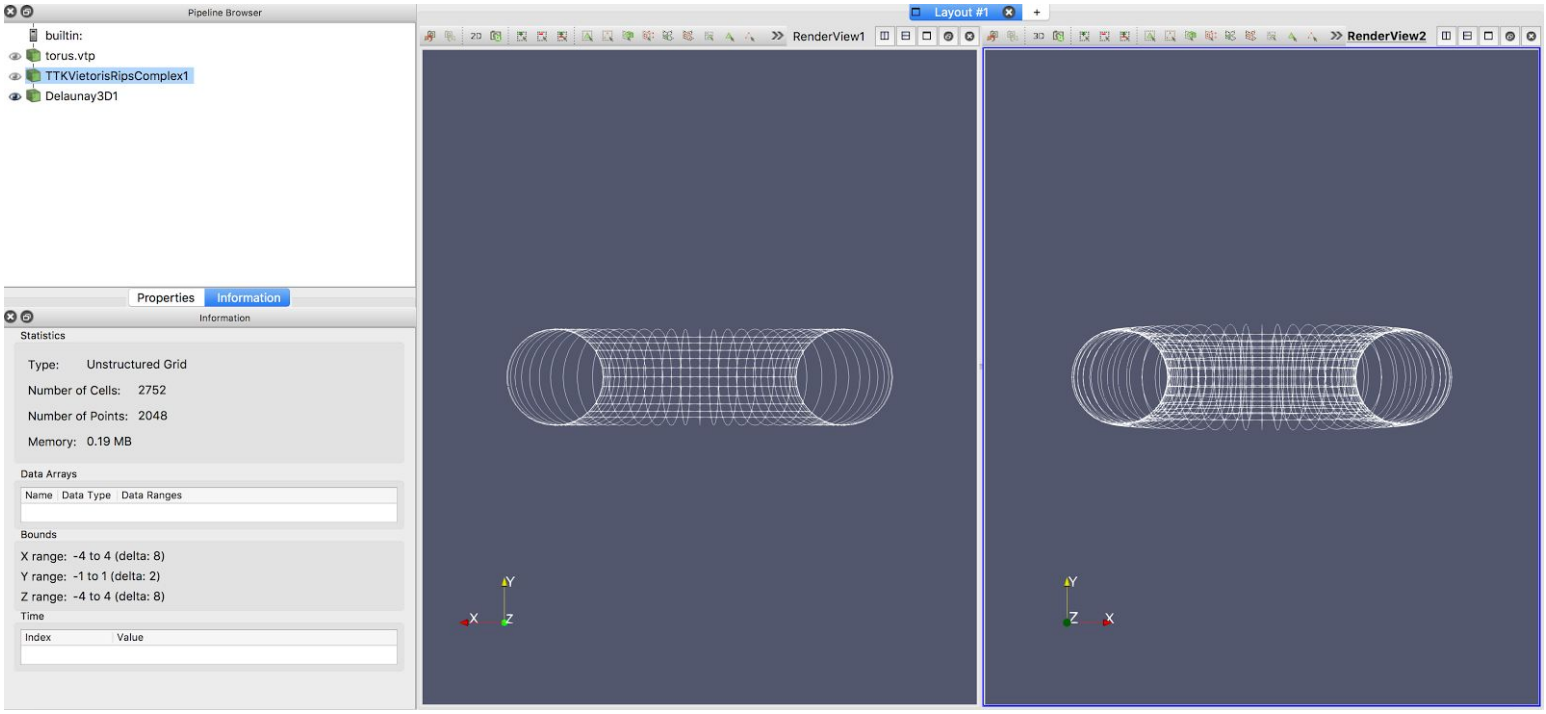


It is interesting to note that border are not line, but actually small triangles close all together. By keep increasing the radius more points starts to connect to each other and this was the result

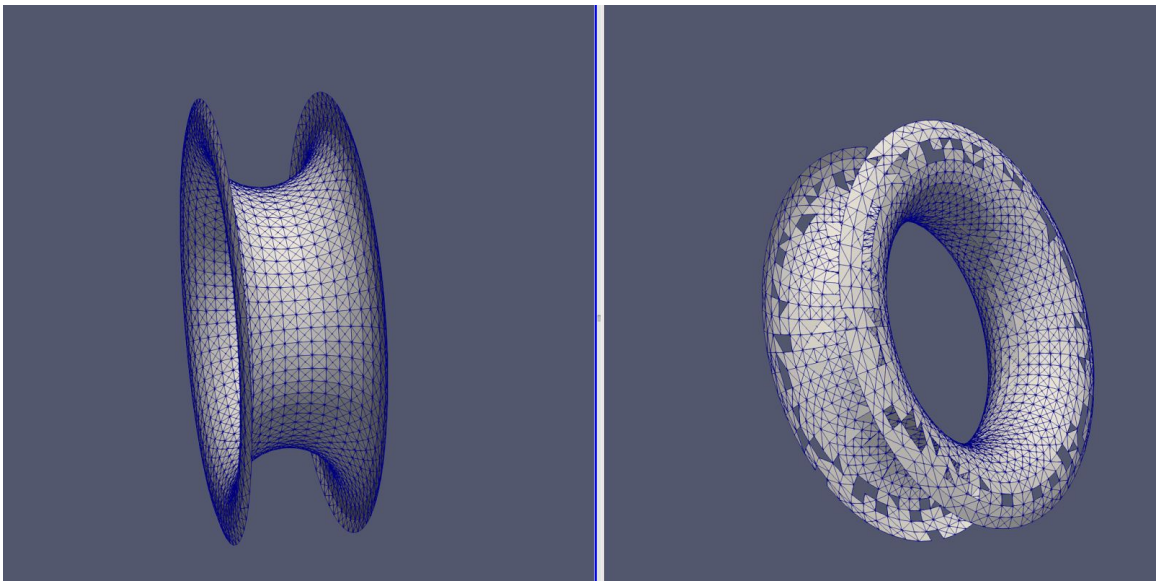


The reason why it is happening because by increasing the radius we are including more combinations and more points to connect so eventually we are going to end up with a blue big disk.

The Delaunay3D plugin is different from the VietorisRipsComplex plugin, since they are not drawing using the methods. However,in my testing testing, for smaller numbers of alpha(for the Delaunay3D plugin) and for the radius (for VietorisRipsComplex plugin), the plugins are the same, visually identical also by taking a look at the properties the number of points and number of cells are the same



However, by increasing the size of the alpha and the radius the difference started to show. In fact by setting the alpha to 0.2 and the same value from the radius we ended up with picture almost identical but the number of cells differs almost 100 more cells for the VietorisRipsComplex plugin. This difference is exponential in fact by kept increasing slightly the number of the radius the difference was almost 2000 more cells for the VietorisRipsComplex plugin



The reason why it happening is because VietorisRipsComplex is using the radius given to determine which edge/triangle/tetra to include, but the Delaunay3D plugin is using alpha to control the output of this filter and only edges, faces, or tetra contained within the circumsphere (of radius alpha) will be output. The Delaunay3d plugin is basically doing a triangulation where a circumsphere of each simplex in a triangulation contains only the n+1 defining points of the simplex. for 3 dimensional simplexes(tetrahedra).

The best possible radius value that creates a spherical shell with no boundary is above 0.151. By looking at the properties we can see that

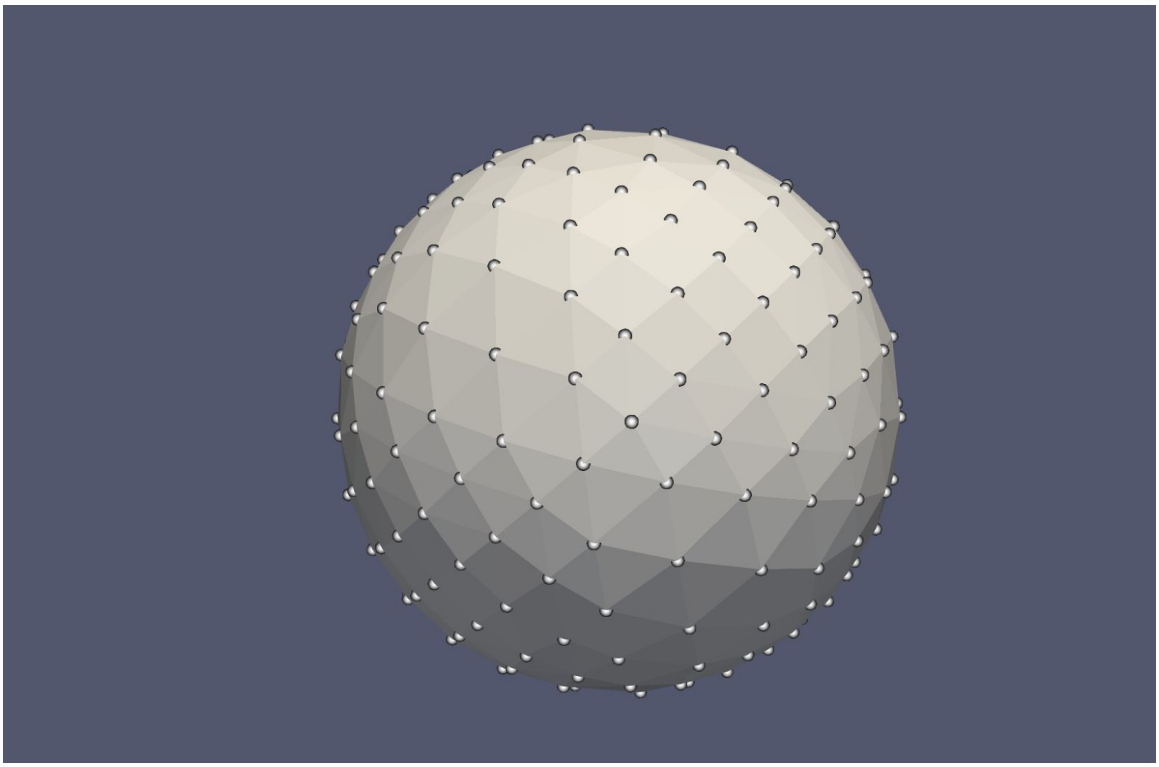
Statistics

Type:	Unstructured Grid
Number of Cells:	560
Number of Points:	258
Memory:	0.045 MB

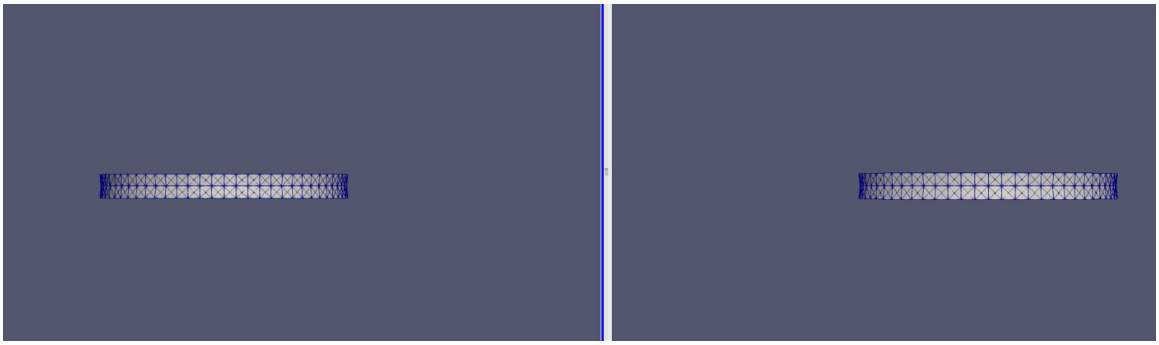
Therefore we can conclude that the euler characteristic:

X = V - E + F

We know that the euler characteristic of a sphere is 2 and by using the extract edges plugin we are able to calculate the number of edges 792 which is close to 818 which is supposed to be the correct number of value in a perfect sphere, but since our sphere is not perfectly smooth I believe we have this difference. The reason why the euler characteristic is 2 is because its genus is 0 since no loops or handles are contained.



By looking at the torus data we can the distance of the points inside the through hole are very close to each other, in fact they are the first to show up with very small radius.



Then gradually by increasing the radius the points outside of the through hole are connected and eventually the entire torus is formed by using the radius above 0.22.

By playing the other 2 data set I immediately understand the this two data set are very big and the size of the radius is very sensitive. In fact the the size of the V-R complex increases rapidly. I believe there are smart ways to optimize this plug-in. I kind use a brute force method by checking all the possible combinations, but I also optimized to check each pair in my edges list every time I have one and if they don't exist I don't bother keep checking the possible combinations with that pair. For example if I have pair 1,2 in my edge list then I will continue to search for pair 1,3 and 2,3. However, if 1,2 is not present in my edge list then I don't check 1,3 and 2,3. I also implemented a truth table for edges for a faster computation to understand if edge is present or not. Another optimization that I can think is to compute the possible combinations by using the list of edges, so you don't have to look to all the possible combination, but just the ones that are present. Yes I believe that the computation can be optimized if we know the radius a priori, for example we can sort the points and use a kd-trees with a search algorithm to achieve the same result. This will make the computation a lot faster